IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named Inventor :	Judy M. Gehman	
Appln. No.:	10/816,213	
Filed :	April 1, 2004	Group Art Unit: 2191
For :	SYSTEM AND METHOD FOR IMPLEMENTING MULTIPLE INSTANTIATED CONFIGURABLE PERIPHERALS IN A CIRCUIT DESIGN	Examiner: Satish Rampuria
Docket No.:	03-1002/L13.12-0246	

BRIEF FOR APPELLANT

FILED ELECTRONICALLY ON June 25, 2009

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This is an appeal from a final rejection of the claims in an Office Action dated October 8, 2008 and the Notice of Panel Decision from Pre-Appeal Brief Review dated June 1, 2009.

REAL PARTY IN INTEREST

LSI Corporation, having offices 1621 Barber Lane, Milpitas, California 95035, has acquired the entire right, title and interest in and to the invention, the application, and any and all patents to be obtained therefore, as set forth in the Assignments recorded on Reel 015590/ Frame 0045 and Reel 020548/ Frame 0977.

RELATED APPEALS AND INTERFERENCES

Applicants are aware of no related appeals or interferences, other than an earlier-filed request for pre-appeal review in the present application.

STATUS OF THE CLAIMS

Claims	Status
None	Withdrawn
None	Canceled
1-20	Rejected
None	Allowed
None	Objected to

Identification of claims being appealed: 1-20

S TATUS OF AMENDMENTS

There are no un-entered amendments filed to the final rejection being appealed.

SUMMARY OF CLAIMED SUBJECT MATTER

A. **Context of Claims**

A designer of large-scale integrated circuits begins with a high-level (abstract) idea of the tasks the system is to perform. To realize the system in some physical technology, the designer determines how to implement the system through the interconnection of millions of transistors to perform the desired operations. (Specification, page 2, lines 5-16).

To address the design time constraints, electronic system design has moved toward a methodology referred to as "block-based design". The block-based design methodology includes a range of techniques that take advantage of existing component design blocks, which have been designed and tested previously. (Specification, page 2, lines 17-28).

But, conventional integrated circuit design methodologies are not well-suited for reuse of pre-designed circuit blocks. . (Specification, page 3, lines 1-9). For example, the Verilog hardware

description standard presented a problem with respect to multiple instantiations of configurable blocks (known as "Intellectual Property" or "IP" Blocks) in an integrated circuit design. Multiple instantiations of a configurable block in a design, where different configurations are used for each instantiation required special handling. In order to handle multiple instantiations of a configurable peripheral or IP block in a chip design, the conventional flow required uniquely naming each peripheral instantiation so that they could co-exist within the same design. (Specification, page 4, line 18 to page 5, line 4).

Within software, multiple instantiations required header files to find register addresses that were renamed to co-exist. In other words, header files/C code were duplicated and renamed. Unfortunately, renaming the peripheral instantiations required the Register Transfer Level code (RTL) to be modified after the design had already been verified to be correct, which potentially introduces errors in the design. Additionally, duplicating and renaming header files required significant coding and debugging time. If the peripheral instantiation is renamed after verification, the "uniquified" RTL code should be verified to ensure that it is equivalent to the original RTL code. However, such a secondary verification process would take away some of the time-to-market benefits of reusable peripheral instantiations. (Specification, page 5, lines 5-22).

For example, within the Verilog hardware description language, the Verilog compiler substitutes the text of the macro wherever it encounters the 'define directive. This substitution is performed for the entire design, so the 'define directive and its associated text macros are global, and are substituted at every level of the design. When a design uses more than one instantiation of a configurable peripheral, the 'define text macros interfere with each other. The compiler will use only one of the definitions for the entire design. (Page 13, line 22 to page 14, line 3).

An embodiment of the present invention is shown in Figure 1, which illustrates a block diagram of a single semiconductor chip 100 synthesized from a hardware description language or RTL description, for example. Chip 100 has pin outs 102, transistor fabric 104, various circuit structures 106, and configurable peripherals 108. In an embodiment of the present invention, the RTL code or HDL code (for example) describes the various structures and interconnections of

chip 100, as well as the configurable peripheral structures 108. During the synthesis process, the code is converted to logical structures for placement on the chip. As shown, more than one instance of a configurable peripheral 108 could be disposed on a single chip 100. (Page 9, line 28 to page 10, line 24).

Each independent claim on appeal is described below with reference to the specification and examples of corresponding elements shown in the figures.

B. **Independent Claim 1**

1. A method for coding a hardware description (See, examples on page 12, line 25 to page 32, line 8) of a peripheral device (FIG. 1, 108; FIG. 3, 316), the method comprising:

configuring (FIG. 5, steps 500, 502; p. 47, ll. 22-28) a function block to instantiate multiple instances of the peripheral device (FIG. 1, 108; FIG. 3, 316) within a single chip design (FIG. 1, 100), the hardware description of the peripheral device having options (FIG. 2, 204; e.g., "parameters" and "define directives" p. 13, ll. 1-5; examples on p. 15, ll. 9 and 21-22; p. 16, ll. 7-12; p. 17, ll. 20-30; p. 18, ll. 6-10; p. 30, ll. 1-28) associated with different configurations of the peripheral device (FIG. 1, 108; FIG. 3, 316);

selecting (FIG. 5, step 504; p. 47, l. 28 to p. 48, l. 1) between the options at compile time (Compile Time 204 in FIG. 2) for each instance of the peripheral device (FIG. 1, 108; FIG. 3, 316) such that at least two of the instances (108, 316) have different configurations from one another, wherein the options are selected without modification to the hardware description (e.g., RTL or HDL code) (e.g., p. 12, l. 25 to p. 13, l. 5; p. 22, ll. 15-20; p. 23, l. 34 to p. 24, l. 7; p. 26, ll. 9-19; p. 27, ll. 10-18; p. 27, l. 34 to p. 28, l. 2; p. 29, ll. 3-8; p. 30, ll. 25-33); and

compiling (FIG. 5, step 504) the hardware description (e.g., RTL or HDL code) to produce a structural model comprising each instance of the peripheral device with the selected options for that instance (p. 4, ll. 1-3; p. 6, ll. 4-26; p. 22, ll. 15-20; p. 23, l. 34 to p.

24, l. 7; p. 26, ll. 9-19; p. 27, ll. 10-18; p. 27, l. 34 to p. 28, l. 2; p. 29, ll. 3-8; p. 30, ll. 25-33; p. 47, ll. 9-14).

Specific examples of configuring different instantiations of a configurable peripheral at compile time for a single integrated circuit design are described on page 12, line 25 to page 32, line 8, for example.

C. **Independent Claim 7**

7. A method for coding a reusable hardware description (See, examples on page 12, line 25 to page 32, line 8) of a peripheral device (FIG. 1, 108; FIG. 3, 316), the method comprising:

configuring (FIG. 5, steps 500, 502; p. 47, ll. 22-28) a function block to instantiate multiple instances of the peripheral device (FIG. 1, 108; FIG. 3, 316) within an integrated circuit design (FIG. 1, 100), the reusable hardware description of the peripheral device having options (FIG. 2, 204; e.g., "parameters" and "define directives" p. 13, ll. 1-5; examples on p. 15, ll. 9 and 21-22; p. 16, ll. 7-12; p. 17, ll. 20-30; p. 18, ll. 6-10; p. 30, ll. 1-28) selectable at compile time (Compile Time 204 in FIG. 2);

instantiating the multiple instances of the peripheral device (FIG. 1, 108; FIG. 3, 316) on the integrated circuit design (FIG. 1, 100) by programmatically selecting (FIG. 5, step 504; p. 47, l. 28 to p. 48, l. 1) between the options at compile time (Compile Time 204 in FIG. 2) for each instance of the peripheral device (FIG. 1, 108; FIG. 3, 316) so that at least two of the instances have different configurations (e.g., p. 22, ll. 15-20; p. 23, l. 34 to p. 24, l. 7; p. 26, ll. 9-19; p. 27, ll. 10-18; p. 27, l. 34 to p. 28, l. 2; p. 29, ll. 3-8; p. 30, ll. 25-33); and

compiling (FIG. 5, step 504) the reusable hardware description (e.g., RTL or HDL code) to produce a structural model comprising the multiple instances of the peripheral device, each with the selected options and resulting configuration for that instance (P. 4, ll. 1-3; p. 6, ll. 4-26; p. 22, ll. 15-20; p. 23, l. 34 to p. 24, l. 7; p. 26, ll. 9-19; p.

27, ll. 10-18; p. 27, l. 34 to p. 28, l. 2; p. 29, ll. 3-8; p. 30, ll. 25-33; p. 47, ll. 9-14).

D. **Independent Claim 16**

16. A method for instantiating multiple instances of a peripheral device (FIG. 1, 108; FIG. 3, 316) within an integrated circuit design (FIG. 1, 100), the method comprising:

configuring (FIG. 5, steps 500, 502; p. 47, ll. 22-28) a hardware description block (e.g., RTL or HDL code) to describe the peripheral device (FIG. 1, 108; FIG. 3, 316) and to describe options associated with different configurations of the peripheral device; and

selecting (FIG. 5, step 504; p. 47, l. 28 to p. 48, l. 1) between the options at compile time (Compile Time 204 in FIG. 2) for each of the multiple instances of the peripheral device without modifying the hardware description block (e.g., p. 12, l. 25 to p. 13, l. 5; p. 22, ll. 15-20; p. 23, l. 34 to p. 24, l. 7; p. 26, ll. 9-19; p. 27, ll. 10-18; p. 27, l. 34 to p. 28, l. 2; p. 29, ll. 3-8; p. 30, ll. 25-33); and

compiling (FIG. 5, step 504) the hardware description (e.g., RTL or HDL code) to produce a structural model comprising the multiple instances of the peripheral device, each instance having the selected options for that instance, wherein the selected options for at least two of the multiple instances are different from one another so that the at least two instances have different configurations (p. 4, ll. 1-3; p. 6, ll. 4-26; p. 22, ll. 15-20; p. 23, l. 34 to p. 24, l. 7; p. 26, ll. 9-19; p. 27, ll. 10-18; p. 27, l. 34 to p. 28, l. 2; p. 29, ll. 3-8; p. 30, ll. 25-33; p. 47, ll. 9-14).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Rejection of claims 1-3, 5-11 and 13-20 under §103(a) based on:

1) Bowen U.S. Publication No. 2002/0100029 in view of

2) Duboc et al. U.S. Patent No. 6,425,116.

2. Rejection of claims 4 and 12 under §103 based on:

1) Bowen U.S. Publication No. 2002/0100029 in view of

2) Duboc et al. U.S. Patent No. 6,425,116 in view of

3) Yu et al., U.S. Patent No. 6,829,754.

ARGUMENT

I. REJECTION OF CLAIMS 1-3, 5-11 and 13-20

Claims 1-3, 5-11 and 13-20 were rejected under §103(a) as being allegedly unpatentable over U.S. Publication No. 2002/0100029 to Bowen in view of U.S. Patent No. 6,425,116 to Duboc et al.

A. Basic Misunderstanding/Mis-Description in Office Action

The claim rejections and various comments provided by the Examiner reflect a basic misunderstanding or misdescription of expressly-recited claim elements.

The analysis provided in the Office Action fails to consider, within the context of the other claim elements, that:

- multiple instances of the same peripheral device are instantiated in a single chip design;
- at least two of the instances within the same chip have different configurations from one another; and
- the different configurations are selected at compile time without modification of the hardware description of the peripheral device.

As a result, the Office Action attempts to combine references that fail to meet the plain language of Applicant's claims.

B. **The Rejection of Independent Claim 1**

1. Bowen

In an effort to support the rejection, the Examiner refers to various paragraphs of Bowen, such as paragraphs [0041], [0009], [0036] and [0138]. These paragraphs use some of the same terms like "compile" and "configuration", but different things are going on.

More specifically, the Office Action incorrectly suggests that Bowen discloses "configuring a function block to instantiate multiple instances of the peripheral device within the single chip . . . the hardware description having options associated with different configurations of the peripheral device . . . wherein the options are selected without modification to the hardware description. . . .."

i. **Bowen Does Not Disclose Different Configurations Instantiated in a Single IC Chip Design**

Bowen teaches using a configuration (without modification to that particular configuration) multiple times in a single chip design. But Bowen does not disclose multiple instances of a peripheral device on the same IC with different configurations, without modification to the hardware description of the peripheral device.

The Examiner mistakenly relies on Bowen paragraph [0241]. In this part of the patent, Bowen discusses a preferred embodiment, which is written using JAVA, C, & C++ languages and uses object oriented programming (OOP) methodology. Paragraph [0241] describes some of the features of OOP, where OOP components are reusable software modules that can be used together.

The Examiner also mistakenly refers to Bowen, paragraphs [0040-0041]. Paragraph [0040] describes taking a behavioral description of the target electronic system and automatically partitioning the required functionality between hardware and software while being able to vary the parameters (size or power) of the hardware and/or software. Varying the parameters of the hardware or software effects the output (the desired netlist or register transfer level (RTL) description). That netlist or register transfer level description (with only a single configuration)

is then used in their FPGA or ASIC [0042]. There is no teaching or suggestion of any further modifications to achieve and instantiate multiple configurations of that netlist or register transfer level description in the same FPGA or ASIC.

Claim 1 requires:

". . . to instantiate multiple instances of the peripheral device within a single chip design . . ."
". . . at least two of the instances have different configurations . . ."; and
". . . produce a structural model comprising each instance of the peripheral device . . ."

Thus, claim 1 requires the structural model to comprise multiple instances of the same peripheral device, each with a different configuration in the same single chip design.

Bowen does not disclose this element or provide any method by which this could be achieved.

ii. **Options are Not Selectable Without Modification To The Hardware Description**

Claim 1 further requires:

". . . such that at least two of the instances have different configurations from one another, wherein the options are selected without modification to the hardware description."

Bowen, states in [0036], "a hardware compiler for producing from those parts of the specification partitioned to hardware a register transfer level description for configuring configurable logic resources."

The register transfer level description configures the configurable logic resources according to the input provided to the hardware compiler. The input provided to the hardware compiler contains a pre-defined configuration that is used for configuring the configurable logic resources, such as during synthesis at step 214 in Figure 2. The input itself is not configurable at compile time. Rather it is predetermined by the C code. Thus, a configuration change would require a change to the hardware description.

The Examiner may be suggesting that the C-like input is equivalent to the claimed "hardware description". However, Bowen discloses creation of multiple RTL descriptions from

the C-like input (it is reusable), whereas, claim 1 allows creation of multiple configurations ("uses") from the same hardware description (e.g., RTL).

As mentioned above, the Examiner refers to Bowen, paragraphs [0040-0041]. Paragraph [0040] describes taking a behavioral description of the target electronic system and automatically partitioning the required functionality between hardware and software while being able to vary the parameters (size or power) of the hardware and/or software. Varying the parameters of the hardware or software effects the output (the desired netlist or register transfer level (RTL) description). That netlist or register transfer level description (with only a single configuration) is then used in their FPGA or ASIC [0042]. There is no teaching or suggestion of any further modifications to achieve multiple configurations of that netlist or register transfer level description in the same FPGA or ASIC.

Bowen Figure 2 shows a C-like input description producing an output "RTL description" (box 212). The "width adjustment" blocks [206] are before block 212 in the flow and therefore before the RTL description is created. As a result, varying the "width adjustment" necessarily modifies the RTL hardware description. This RTL description is then targeted to an FGPA/ASIC [0144, 0145]. The FPGA/ASIC is programmed with a single RTL configuration.

Thus, it is incorrect to state that Bowen does discloses options selected at compile time "without modification to the hardware description".

iii. **Bowen Does Not Disclose Selecting Between the Options at Compile Time for Each Instantiation of the Peripheral Device**

The Office Action acknowledges that Bowen does not disclose "selecting between the options at compile time for each instance of the peripheral device such that at least two of the instances have different configurations from one another," as recited in claim 1.

2. Duboc

Doboc et al. generally discloses a way to take existing building blocks and user input to make RTL (a hardware description), reduce to gates, and make an integrated circuit.

Duboc et al. does not disclose using the same building block more than once with different configurations in the same IC. Duboc mentions reusable templates, nowhere in Duboc does it say the same template is used multiple times with different inputs (configurations) in the same DSP.

i. **Duboc et al. Fails to Disclose That Two Different Instantiations Can Have Two Different Configurations Selectable at Compile Time**

The Examiner refers to col. 5, line 54 to col. 6, line 2, but this paragraph does not say two custom DSPs (or customizable circuit blocks with different configurations) are produced in the same IC. Duboc simply discloses that a custom DSP is produced (meaning one configuration).

Further, Duboc's Col. 3, lines 7-15 do not say that different configurations are used in the same DSP integrated circuit that is produced.

The Office Action incorrectly states Duboc et al. shows "selecting between the options at compile time for each instance of the peripheral device such that at least two of the instances have different configurations from one another," citing Duboc, col. 8, lines 33-39; col. 10, lines 31-34.

Column 8, lines 33-39 mentions using the GUI input to initiate generation of the IC design via selection of a "compile option" from the GUI window, resulting in the execution of a script engine that verifies parameters input by a user.

In Duboc et al., the user configures the IC once. In the example described in column 6, lines 47-57, they are making one custom DSP IC. Looking at the options of Table 1, the user picks one Data Y ROM size to make this one custom DSP IC. The user does not run the GUI (or a template) twice to have two or more custom DSP configurations (different DSP instances) on one IC. Rather, the user runs through the GUI once and it configures the IC.

For example, there is no an instance identifier in FIG. 6 (so as to independently configure different instances in different ways). The Y RAM and X RAM are tied together with the same

configuration (their address space is divided). "The Y data space, which is configurable by the user, can occupy the top 2, 4, 8 or 16K of the data space, with the remaining data addresses mapped to the X data space." (Col. 6, lines 47-57; see also, col. 9, lines 43-47). They are not separately configurable instances of the same peripheral device.

In the present application, the same building block (peripheral) can be used twice in the same IC and have a different configuration. So, for example, the user could configure the Data Y ROM to be 2K in one custom DSP instance on the IC and the Data Y ROM to be 4K in another/different custom DSP instance on the same IC.

Further, the Examiner mistakenly refers to Duboc, col. 10, lines 31-34, "generates models for customized memory components suitable of interfacing within custom DSP integrated circuit." However, the DSP integrated circuit is custom from other DSP ICs, but does not include multiple instances of a separately customized DSPs (with different options) or of the same memory device (with different options) on the same IC.

Duboc et al. do not disclose that two different instances of the same peripheral device can have two different configurations selectable at compile time.

Referring to the language of claim 1, Duboc et al. do not disclose "configuring a function block to instantiate multiple instances of the peripheral device within a single chip design, the hardware description of the peripheral device having options associated with different configurations of the peripheral device; [and] selecting between the options at compile time for each instance of the peripheral device such that at least two of the instances have different configurations from one another, wherein the options are selected without modification to the hardware description."

In fact, Duboc et al. do not suggest that such a feature would be desirable or how such a feature could be accomplished. The disclosed GUI does not provide that structure or functionality.

3. Combination of Bowen and Duboc et al.

Lacking such a disclosure, the proposed combination of Bowen and Duboc et al. therefore fails to teach or suggest multiple instances of a peripheral device on the same IC with different configurations, where the same function block is used to instantiate a hardware description with options associated with the different configurations of the peripheral device.

The proposed combination also fails to disclose a step of selecting between the options at compile time for each instance of the peripheral device without modification to the hardware description.

Further, there is nothing in either Bowen or Duboc et al. that would make it obvious for a skilled person to try.

C. **Dependent Claims 2 and 3 (similarly claims 15, 17, 20)**

Claim 2 adds that the step of selecting comprises "passing a parameter value to the function block at compile time for each instance of the hardware peripheral."

Applicant does not understand why the Examiner cited Bowen paragraph [0111] or how it applies to claim 2. This paragraph refers to the user defining the scheduling decisions (operations happen on certain clock cycles) to the compiler. This is the compiler that takes the C code (or RTL language) and creates the hardware description to put into the FPGA. (Bowen para. [0105]).

Claims 3 describes that the configuration options are peripheral design functions, peripheral design pin widths, or peripheral design interface pin outs.

Again, neither Bowen nor Duboc et al. disclose a method in which a parameter value can be passed to a function block that is configured to instantiate each instance of a hardware peripheral, where the different instances can have different configurations (per claim 1).

The Office Action cites Bowen paragraph [0037], which states that "The system can include a width adjuster for setting and using a desired data word size, and this can be done at several points in the desired process as necessary."

But in Bowen, the same width adjustment would be applied to every instance in the

design. Paragraph [0114] describes that at 206a and 206b, width adjustment is carried out by ANDing bits with a bit mask. This would apply to every instance of the block/code in the design.

In the present application, the functional block can be configured to use a different width for each instance of the block in the design, for example.

D. **The Rejection of Independent Claim 7**

In respect to independent claim 7, the Office Action refers to paragraphs [0009], [0031] and [0241] of Bowen. Paragraph [0241] simply states that the OOP components (reusable software modules) are accessed at run-time.

As discussed above, Bowen does not disclose configuring a function block to instantiate a hardware description with options at compile time and instantiating multiple instances of the peripheral device on the integrated circuit by programmatically selecting between the options at compile time for each instance of the peripheral device, as recited by claim 7.

As described above, Duboc et al. also fails to disclose these elements.

Claim 7 and its respective dependent claims are therefore not obvious over Bowen in view of Duboc et al. for similar reasons as were discussed above with respect to claim 1.

E. **The Rejection of Independent Claim 16**

Similarly, independent claim 16 and its dependent claims are not obvious over Bowen in view of Duboc et al. for the similar reasons as were discussed above with respect to claims 1 and 7.

F. **No *Prima-Facie* Case For Obviousness**

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally,

the prior art reference (or references when combined) must teach or suggest all of the claim limitations. In re Vaeck, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991).

The initial burden is on the examiner to provide some suggestion of the desirability of doing what the inventor has done. "To support the conclusion that the claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references." Ex parte Clapp, 227 USPQ 972, 973 (Bd. Pat. App. & Inter. 1985).

Under the criteria set forth in Vaeck, the final Office Action fails to establish a *prima facie* case of obviousness of claims 1-3, 5-11, and 13-20 based on Bowen and Duboc. Further, the combination of cited references fail to describe, expressly or impliedly, all of the elements recited in claims 1-3, 5-11, and 13-20, particularly since both references fail to disclose similar elements of Applicant's claims. Moreover, the Examiner has not presented a convincing line of reasoning as to why an artisan would have found the claimed invention to have been obvious in light of the teachings of the references.

Applicant therefore respectfully requests that the rejections of claims 1-3, 5-11 and 13-20 under §103(a) based on Bowen and Duboc be reversed.

II. CLAIM REJECTIONS UNDER §103 BASED ON BOWEN, DUBOC ET AL. AND YU ET AL. SHOULD BE REVERSED

Claims 4 and 12 were rejected under §103 as being allegedly being unpatentable over Bowen in view of Duboc et al. and Yu et al., U.S. Patent No. 6,829,754.

Claim 4 (and similarly claim 12) further requires that "selecting between options at compile time" comprises: "tying strap pins to power or ground".

The combination of Bowen, Duboc et al. and Yu et al. does not teach or make obvious the inventions recited in claims 4 and 12 for similar reasons as discussed above with respect to the independent claims.

In addition, page 17 of the Office Action suggests with regard to Yu, "one of ordinary skill in the art would be motivated to strap the power or ground pins so that the power related problems can be avoid[ed]".

The strap pins discussed in claims 4 and 12 are not for power related problems. They are to tie peripheral input pins to a logic high or logic low level (based on the particular configuration of that instance) to make a functional decision on different features supported in a design. For example, processors and computer systems have the concept of big endian data format and little endian data format. These formats specify how bytes are organized in the memory. A peripheral can have a static strap pin tied to ground (low) if little endian data format should be used or to power (high) if big endian data format is used. The pin is tied high or low during configuration.

The comment the Examiner makes about power related problems and the citation to Yu column 11, lines 2-5 relate to current flow or power through an actual piece of metal and generating a warning if the physical width of the strap is smaller than the power pin to which it connects. This has nothing to do with configuring a functional block to tie a strap pin to power or ground for implementing a particular configuration of a peripheral device.

Similarly, Yu column 3, lines 18-35 refer to the use of flow straps, slot straps and pad straps in relation to power distribution and not in relation to a logic or functional change in the ASIC design.

Again, this has nothing to do with configuring a functional block to tie a strap pin to power or ground for implementing a particular configuration of a peripheral device.

Bowen in view of Duboc et al. and Yu et al. therefore do not support the rejection of claims 2 and 12. The rejection of claims 2 and 12 should therefore be reversed.

III. CONCLUSION

Applicants respectfully request that the rejection of the pending claims under §103(a) be reversed.

The Director is authorized to charge any fee deficiency required by this paper or credit any overpayment to Deposit Account No. 23-1123.

WESTMAN, CHAMPLIN & KELLY, P.A.

By: /David D. Brush/
David D. Brush, Reg. No. 34,557
Suite 1400 - International Centre
900 Second Avenue South
Minneapolis, Minnesota 55402-3319
Phone: (612) 334-3222 Fax: (612) 334-3312

DDB:

CLAIMS APPENDIX

Claims involved in the Appeal:

1. (Previously Presented) A method for coding a hardware description of a peripheral device, the method comprising:

configuring a function block to instantiate multiple instances of the peripheral device within a single chip design, the hardware description of the peripheral device having options associated with different configurations of the peripheral device;

selecting between the options at compile time for each instance of the peripheral device such that at least two of the instances have different configurations from one another, wherein the options are selected without modification to the hardware description; and

compiling the hardware description to produce a structural model comprising each instance of the peripheral device with the selected options for that instance.

2. (Previously Presented) The method of claim 1 wherein the step of selecting comprises:

passing a parameter value to the function block at compile time for each instance of the hardware peripheral; and

instantiating the peripheral device using code according to the parameter value.

3. (Previously Presented) The method of claim 1 wherein the configuration options comprise at least one of peripheral design functions, peripheral design pin widths, or peripheral design interface pin outs.

4. (Previously Presented) The method of claim 1 wherein selecting between options at compile time comprises:

tying strap pins to power or ground.

5. (Original) The method of claim 1 wherein the step of configuring comprises:

configuring the function block with local runtime constants adapted to be overridden individually at compile time.

6. (Original) The method of claim 5 wherein the step of selecting comprises

overriding selected runtime constants at compile time to select between the variable options for each instance of the peripheral device.

7. (Previously Presented) A method for coding a reusable hardware description of a peripheral device, the method comprising:

configuring a function block to instantiate multiple instances of the peripheral device within an integrated circuit design, the reusable hardware description of the peripheral device having options selectable at compile time;

instantiating the multiple instances of the peripheral device on the integrated circuit design by programmatically selecting between the options at compile time for each instance of the peripheral device so that at least two of the instances have different configurations; and

compiling the reusable hardware description to produce a structural model comprising the multiple instances of the peripheral device, each with the selected options and resulting configuration for that instance.

8. (Original) The method of claim 7 wherein the variable options are selected without modification to the reusable hardware description.

9. (Previously Presented) The method of claim 7 wherein the step of configuring comprises:

adding one or more peripheral devices based on desired features of the reusable hardware to the integrated circuit design at compile time.

10. (Previously Presented) The method of claim 7 wherein the step of configuring comprises:

instantiating peripheral devices onto the integrated circuit according to the reusable hardware description wherein the configuration of each instance is unique based on a design parameter.

11. (Original) The method of claim 10 wherein the design parameter comprises a signal width of the peripheral device.

12. (Previously Presented) The method of claim 7 wherein instantiating further comprises:

selecting between the options to define further the function block by tying strap pins to ground or to power.

13. (Original) The method of claim 7 wherein the step of configuring further comprises:

configuring the function block with parameters local in scope, the parameters adapted to be overridden individually at compile time.

14. (Original) The method of claim 13 wherein the step of selecting comprises

overriding selected runtime constants at compile time to select between the options for each instance of the peripheral device.

15. (Previously Presented) The method of claim 7 wherein the step of configuring comprises:

passing a parameter value to the function block at compile time for each instance of the peripheral device; and

instantiating the peripheral device using the reusable hardware description according to the parameter value.

16. (Previously Presented) A method for instantiating multiple instances of a peripheral device within an integrated circuit design, the method comprising:

configuring a hardware description block to describe the peripheral device and to describe options associated with different configurations of the peripheral device; and

selecting between the options at compile time for each of the multiple instances of the peripheral device without modifying the hardware description block; and

compiling the hardware description to produce a structural model comprising the multiple instances of the peripheral device, each instance having the selected options for that instance, wherein the selected options for at least two of the multiple instances are different from one another so that the at least two instances have different configurations.

17. (Previously Presented) The method of claim 16 wherein the step of selecting comprises:

passing a parameter value to the function block at compile time for each instance of the hardware peripheral; and

instantiating the peripheral device with options determined by the parameter value.

18. (Original) The method of claim 16 wherein the step of configuring comprises:

coding the hardware description block with local runtime constants adapted to be overridden individually at compile time.

19. (Original) The method of claim 16 wherein the variable options comprise local runtime constants and wherein the step of selecting comprises:

selecting one or more of the local runtime constants at compile time; and

overriding the selected one or more of the local runtime constants to differentiate each instance of the peripheral device as needed.

20. (Previously Presented) The method of claim 16 wherein the options comprise at least one of peripheral device functions, peripheral device pin widths or peripheral device signal widths.

EVIDENCE APPENDIX

None.

RELATED PROCEDINGS APPENDIX

There are no known related appeals or interferences that will directly affect or be directly affected by or have a bearing on the Board's decision in this Appeal.